

```

2  # 1. Définir MAX_INVEST comme 500 * 100. Cela représente le budget maximal que nous pouvons investir.
3  import csv
4  from itertools import combinations
5  MAX_INVEST = 500 * 100
6
7
8  # 2. Définir une fonction get_csv_data:
9  def get_csv_data():
10     # - Ouvrir le fichier brutforce.csv pour la lecture.
11     with open("data/brutforce.csv", newline="") as csv_file:
12         csv_reader = csv.reader(csv_file, delimiter=",")
13         next(csv_reader)
14         # - Pour chaque ligne du fichier (en ignorant la première ligne qui est l'en-tête):
15         for row in csv_reader:
16             # - Extraire le nom de l'action, le prix et le bénéfice.
17             stock_name = row[0]
18             # - Convertir le prix et le bénéfice de l'euro en centimes.
19             price_in_cents = float(row[1]) * 100
20             benefit_in_cents = float(row[2]) * 100
21             # - Renvoyer le nom de l'action, le prix en centimes et le bénéfice en centimes comme une ligne de données.
22             yield (stock_name, int(price_in_cents), int(benefit_in_cents))
23
24
25  # 3. Définir une fonction generate_combinations:
26  def generate_combinations(stocks):
27     # - Initialiser une variable profit à 0. Cela représente le profit maximal actuel.
28     profit = 0
29     # - Initialiser une liste best_combination vide. Cela représente la meilleure combinaison d'actions actuelle.
30     best_combination = []
31     # - Pour chaque sous-ensemble d'actions (obtenues en utilisant l'algorithme de combinaisons, de taille 1 à la
32     #   taille totale des actions):
33     for i in range(len(stocks)):
34         list_combinations = combinations(stocks, i + 1)
35         for combination in list_combinations:
36             # - Calculer le coût total de la combinaison d'actions.
37             total_cost = sum([stock[1] for stock in combination])
38             # - Si le coût total est inférieur ou égal à MAX_INVEST :

```

```

39         if total_cost <= MAX_INVEST:
40             # - Calculer le profit total de la combinaison d'actions.
41             total_profit = sum([stock[2] for stock in combination])
42             # - Si le profit total est supérieur au profit maximal actuel :
43             if total_profit > profit:
44                 # - Mettre à jour le profit maximal et la meilleure combinaison d'actions.
45                 profit = total_profit
46                 best_combination = combination
47     return best_combination
48
49
50 # 4. Définir une fonction display_result :
51 def display_result(best_combination):
52     # - Afficher le nom de chaque action, son prix et son bénéfice dans la meilleure combinaison d'actions.
53     print("Liste des actions achetées :\n")
54     for stock in best_combination:
55         print(f"{stock[0]} {stock[1] / 100}€ {stock[2] / 100}€")
56     # - Calculer et afficher la somme dépensée et le profit total.
57     print(f"\nSomme dépensée : {sum([stock[1] for stock in best_combination]) / 100}€")
58     print(f"Profit total : {sum([stock[2] for stock in best_combination]) / 100}€")
59
60
61 # 5. Si le script est exécuté en tant que programme principal :
62 if __name__ == "__main__":
63     # - Appeler la fonction get_csv_data pour obtenir les actions.
64     stocks = [stock for stock in get_csv_data()]
65     # - Appeler la fonction generate_combinations pour obtenir la meilleure combinaison d'actions.
66     best_combination = generate_combinations(stocks)
67     # - Appeler la fonction display_result pour afficher le résultat.
68     display_result(best_combination)
69

```

● devpyt@WALIDANOVSKY:~/Bureau/Projet-7\$ python3 brutforce.py
Liste des actions achetées :

Action-4	70.0€	14.0€
Action-5	60.0€	10.19€
Action-6	80.0€	20.0€
Action-8	26.0€	2.86€
Action-10	34.0€	9.18€
Action-11	42.0€	7.14€
Action-13	38.0€	8.74€
Action-18	10.0€	1.4€
Action-19	24.0€	5.04€
Action-20	114.0€	20.52€

Somme dépensée : 498.0€
Profit total : 99.07€

○ devpyt@WALIDANOVSKY:~/Bureau/Projet-7\$